0360-1

Box 1/SEL

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Applicant:

Porat Erlich

Serial No.: 09/820,629

Filed: March 30, 2001

For: METHOD AND SYSTEM FOR EVALUATION OF ELECTRICAL CONDUCTIVITY OF DNA SEQUENCES

§ Group Art Unit: 2857

§ Attorney Docket: 2164/4

Initial Patent Examination Division
Commissioner of Patents and Trademarks
Washington, D.C. 20231

CORRECTION OF APPLICATION PAPERS

Sir:

This is in response to the Notice to File Corrected Application Papers mailed May 2, 2001, which response is being made on or before July 2, 2001, and for which no extension fees are due.

Substitute drawings in compliance with 37 CFR 1.84 are enclosed.

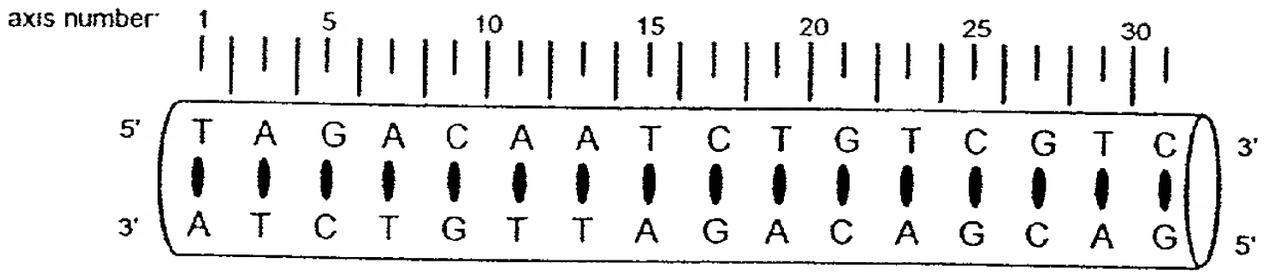
Attached is an ititial paper copy of a "Sequence Listing" for the sequence presented in the current application if Figure 1A, as well as an initial computer form copy of the "Sequence Listing". Applicant respectfully requests the said "Sequence Listing" be entered into the currect application. The content of the sequence listing information recorded in computer readable form is identical to the written sequence listing and includes no new matter.

Respectfully submitted,

Mark M. Friedman
Attorney for Applicant
Registration No. 33,883

Date: June 17, 2001

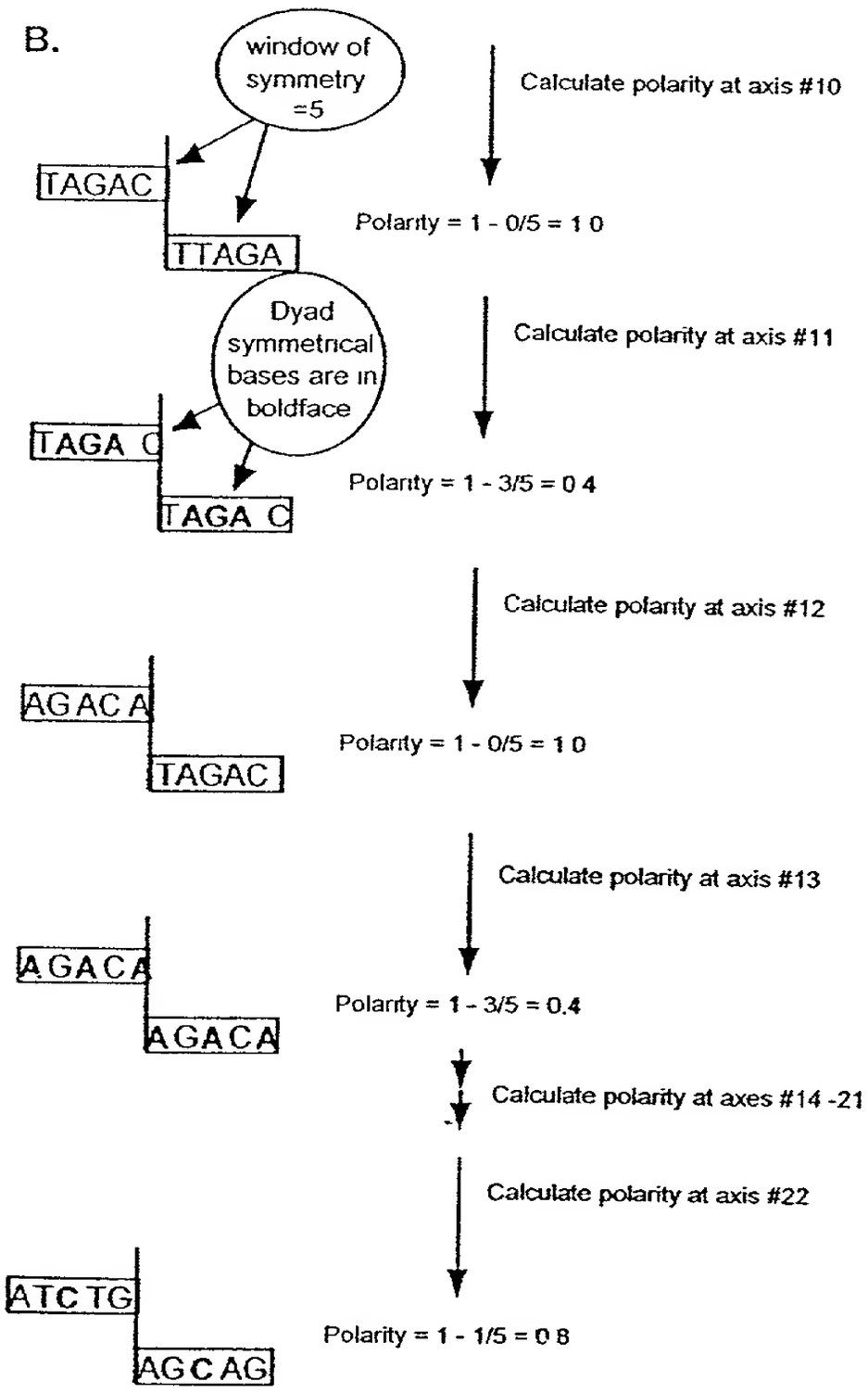
A.



C.

Axis number	Polarity
10	1
11	0 4
12	1
13	0 4
14	0 8
15	0 2
16	0 8
17	0 8
18	0 6
19	0 8
20	0.8
21	0 8
22	0 8
Average	0 708

B.



D.

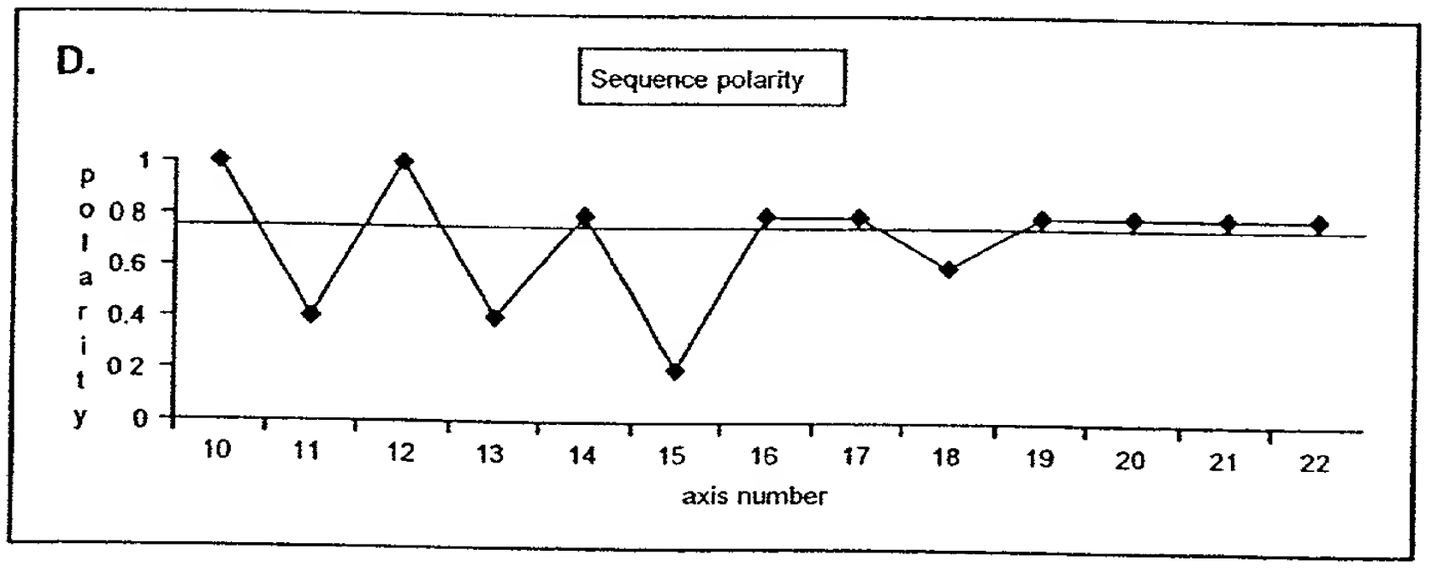


FIGURE 1

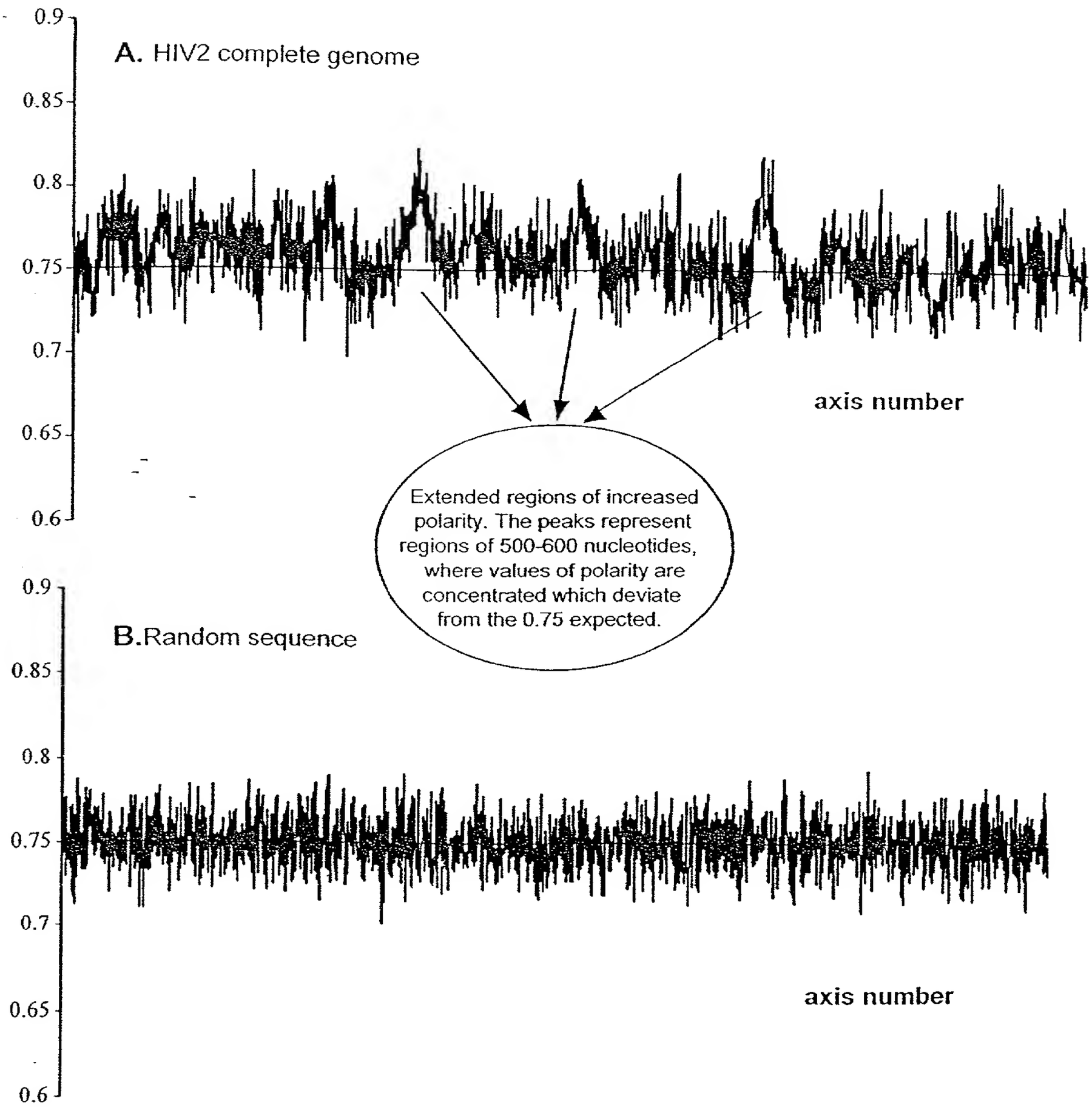
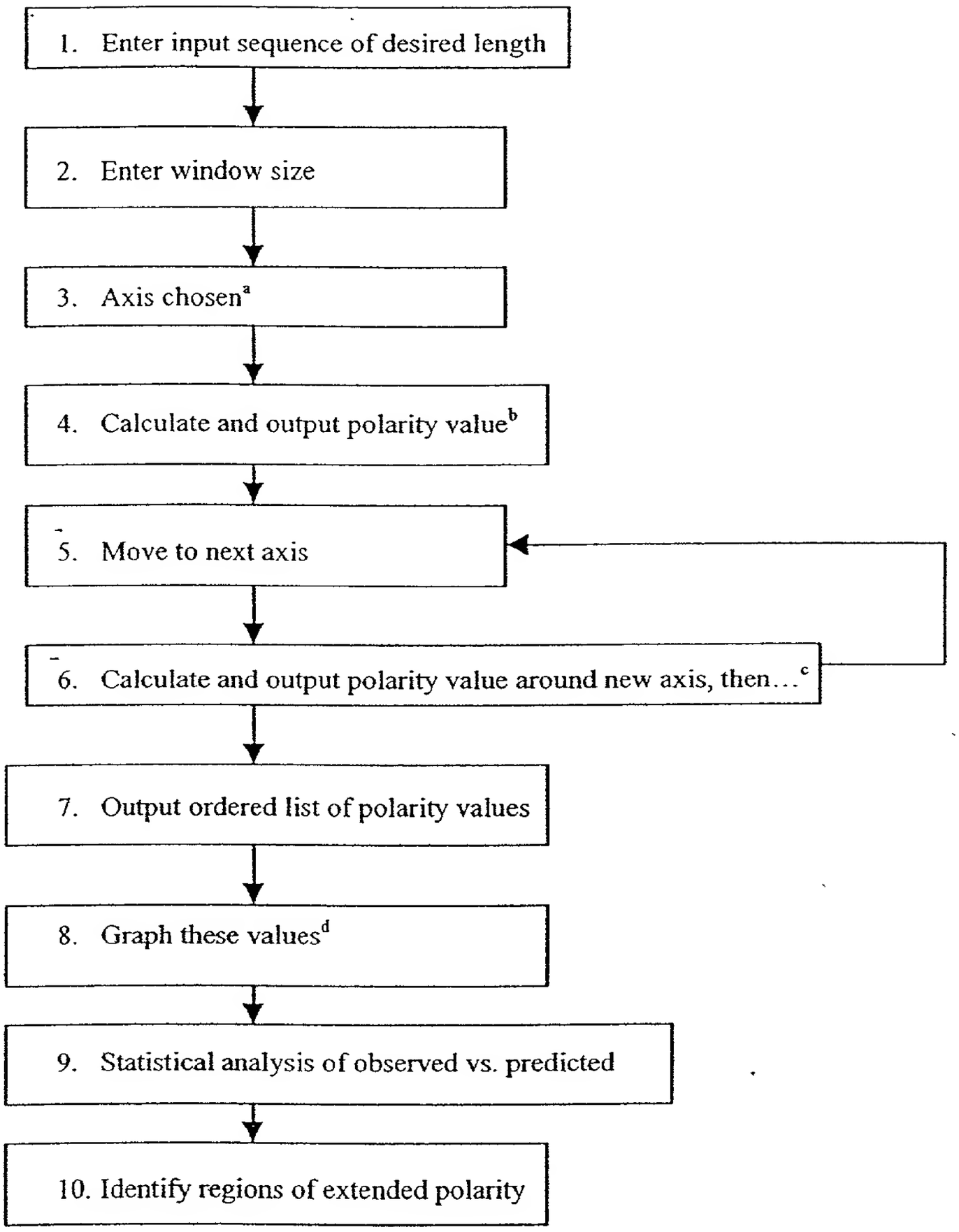

A. HIV2 complete genome
0.9
0.85
0.8
0.75
0.7
0.65
0.6
axis number
Extended regions of increased polarity. The peaks represent regions of 500-600 nucleotides, where values of polarity are concentrated which deviate from the 0.75 expected.
B.Random sequence
0.9
0.85
0.8
0.75
0.7
0.65
0.6
axis number

FIGURE 2



[a] Starting at position = (2*window of symmetry)
[b] [1-(S/W)]
[c] Up to and including axis position = [2*length – (2*window size)]
[d] Can use a moving average of values (with number of values averaged and increment of moving being variable) to smooth curve

FIGURE 3

The algorythm was implemented in PERL programing language.
PERL variable-names and function-names are in boldface.

1 import input sequence as a string variable **$input_seq**

2 prompt user for (**$win_sym**) length

3 cut a length of (2***$win_sym**) bases from the 5' end of**$input_seq** and assign this substring into **$win_seq**

perform (2*win_sym) iterations:
- **chop** (cut last character off a string and return it**$win_seq**
- **unshift** chopped character (prepend it to the front of an array, moving all elements one step to the right) into an initially empty indexed array **@trgt_fwd**)

translate **$win_seq** so that every G or A or T or C of the original string become C or T or A or G respectively of the translated string

perform (2*win_sym) iterations:
- **chop** the translated **$win_seq**
- **push** chopped character (stack it onto the end of an array, without moving the other elements) onto an initially empty indexed array **@trgt_revcomp**)

4 assign **$match_count = 0**

assign index variable **$i = 1**

perform (win_sym) iterations:
- compare between the **$i** th elements of arrays**@trgt_fwd** and **@trgt_revcomp** ; if equal then increase **$match_count** by 1, else nothing.
- increase **$i** by 1

calculate **$asym_count = 1 - ($match_count / win_sym)**

push $asym_count onto initially empty indexed array **@axis_list**

5 while **$input_seq** is not empty

cut one base off of the 5' end of **input_seq** and assign it into**$basefeed**

translate $basefeed GATC->CTAG respectively and assign the translated character into**$basefeed_comp**

unshift $basefeed_comp (prepend it to the front of an array, moving all elements one step to the right) into array**@trgt_revcomp**

pop (remove last element of an array)**@trgt_revcomp**

6 assign **$match_count = 0**

assign index variable **$i = 1**

perform (win_sym) iterations:
- compare between the **$i** th elements of arrays**@trgt_fwd** and **@trgt_revcomp** ; if equal then increase **$match_count** by 1, else nothing.
- increase **$i** by 1

calculate **$asym_count = 1 - ($match_count / win_sym)**

push $asym_count onto **@axis_list**

5' **shift** (remove first element of an array and move all elements one step leftward) **@trgt_fwd**

push $basefeed onto array**@trgt_fwd**

6' assign **$match_count = 0**

assign index variable **$i = 1**

perform (win_sym) iterations:
- compare between the **$i** th elements of arrays**@trgt_fwd** and **@trgt_revcomp** ; if equal then increase **$match_count** by 1, else nothing.
- increase **$i** by 1

calculate **$asym_count = 1 - ($match_count / win_sym)**

push $asym_count onto **@axis_list**

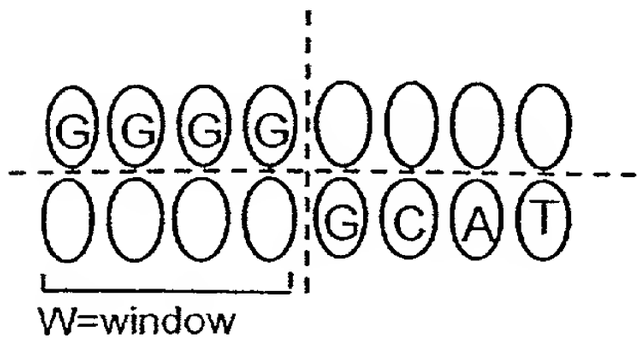
7 save **@axis_list** to file

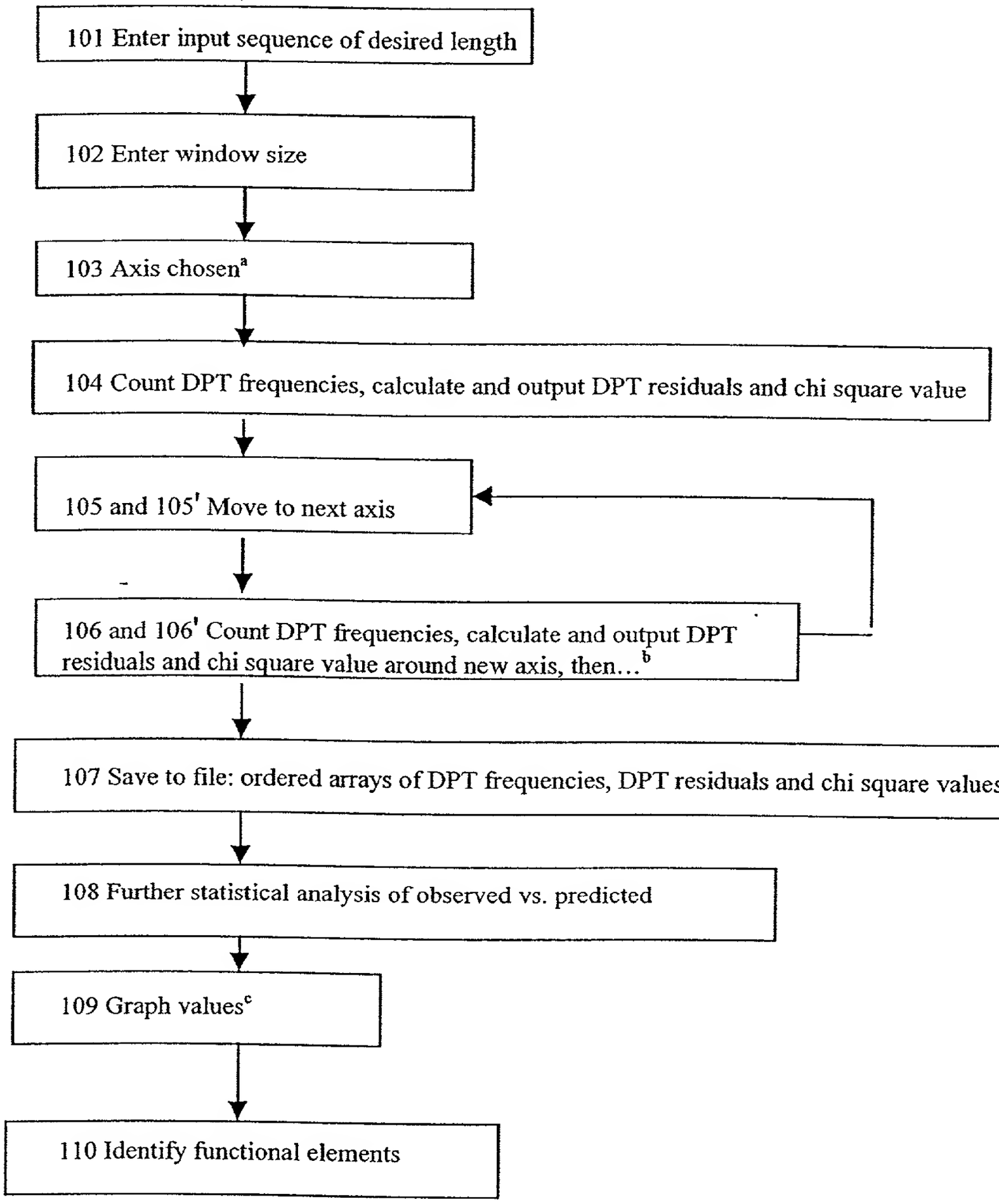
FIGURE 4

self dyad	self mirror	Purine pyrimidine dyad	Purine pyrimidine mirror
i=1: $^{G}_{--G}$	i=5: $^{G}_{--C}$	i=9 : $^{G}_{--A}$	i=13: $^{G}_{--T}$
i=2: $^{A}_{--A}$	i=6: $^{A}_{--T}$	i=10: $^{A}_{--G}$	i=14: $^{A}_{--C}$
i=3: $^{T}_{--T}$	i=7: $^{T}_{--A}$	i=11: $^{T}_{--C}$	i=15: $^{T}_{--G}$
i=4: $^{C}_{--C}$	i=8: $^{C}_{--G}$	i=12: $^{C}_{--T}$	i=16: $^{C}_{--A}$

Figure 5A

Figure 5B





[a] Starting at axis position = (2*window size)

[b] Up to and including axis position = [2*length – (2*window size)]

[c] Values include DPT frequencies, statistical measures including residuals and χ^2

FIGURE 6

The algorythm was implemented in PERL programming language.
PERL variable-names and function-names are in boldface.

101 import input sequence as a string variable ($input_seq)

102 prompt user for ($win_sym) length

103 cut a length of (2*$win_sym) bases from the 5' end of $input_seq and assign this substring into $win_seq

perform (2*win_sym) iterations:
- chop (cut last character off a string and return it) $win_seq
- unshift chopped character (prepend it to the front of an array, moving all elements one step to the right) into an initially empty indexed array (@trgt_fwd)

translate $win_seq so that every G or A or T or C of the original string become C or T or A or G respectively of the translated string

perform (2*win_sym) iterations:
- chop the translated $win_seq
- push chopped character (stack it onto the end of an array, without moving the other elements) onto an initially empty indexed array (@trgt_revcomp)

104 assign $gg_count = 0, $ga_count = 0 ... one such variable for each of the 16 DPTs

assign index variable $i = 1

perform (win_sym) iterations:
- if the $i th elements of arrays @trgt_fwd and @trgt_revcomp are G and G, respectively, then increase $gg_count by 1. Repeat this conditional operation for each of the 16 possible DPTs.
- increase $i by 1

calculate residuals $gg_res = $gg_count - ((1/16) x $win_sym) Repeat this calculation for each of the 16 possible DPTs

push $gg_res onto @gg_res Repeat this step for each of the 16 possible DPTs.

Calculate chi square. $chi_sq = (($gg_res **2) / ((1/16) x $win_sym) +
($ga_res **2) / ((1/16) x $win_sym) +
($gt_res **2) / ((1/16) x $win_sym) +
($gc_res **2) / ((1/16) x $win_sym) +
($ag_res **2) / ((1/16) x $win_sym) +
($aa_res **2) / ((1/16) x $win_sym) +
($at_res **2) / ((1/16) x $win_sym) +
($ac_res **2) / ((1/16) x $win_sym) +
($tg_res **2) / ((1/16) x $win_sym) +
($ta_res **2) / ((1/16) x $win_sym) +
($tt_res **2) / ((1/16) x $win_sym) +
($tc_res **2) / ((1/16) x $win_sym) +
($cg_res **2) / ((1/16) x $win_sym) +
($ca_res **2) / ((1/16) x $win_sym) +
($ct_res **2) / ((1/16) x $win_sym) +
($cc_res **2) / ((1/16) x $win_sym))

push $chi_sq onto @chi_sq

105 while $input_seq is not empty

cut one base off of the 5' end of input_seq and assign it into $basefeed

translate $basefeed GATC->CTAG respectively and assign the translated character into $basefeed_comp

unshift $basefeed_comp (prepend it to the front of an array, moving all elements one step to the right) into array @trgt_revcomp

pop (remove last element of an array) @trgt_revcomp

106 assign $gg_count = 0, $ga_count = 0 ... one such variable for each of the 16 DPTs

assign index variable $i = 1

perform (win_sym) iterations:
- if the $i th elements of arrays @trgt_fwd and @trgt_revcomp are G and G, respectively, then increase $gg_count by 1. Repeat this conditional operation for each of the 16 possible DPTs.
- increase $i by 1

calculate residuals $gg_res = $gg_count - ((1/16) x $win_sym) Repeat this calculation for each of the 16 possible DPTs

push $gg_res onto @gg_res Repeat this step for each of the 16 possible DPTs

Calculate chi square. $chi_sq = (($gg_res **2) / ((1/16) x $win_sym) +
($ga_res **2) / ((1/16) x $win_sym) +
($gt_res **2) / ((1/16) x $win_sym) +
($gc_res **2) / ((1/16) x $win_sym) +
($ag_res **2) / ((1/16) x $win_sym) +
($aa_res **2) / ((1/16) x $win_sym) +
($at_res **2) / ((1/16) x $win_sym) +
($ac_res **2) / ((1/16) x $win_sym) +
($tg_res **2) / ((1/16) x $win_sym) +
($ta_res **2) / ((1/16) x $win_sym) +
($tt_res **2) / ((1/16) x $win_sym) +
($tc_res **2) / ((1/16) x $win_sym) +
($cg_res **2) / ((1/16) x $win_sym) +
($ca_res **2) / ((1/16) x $win_sym) +
($ct_res **2) / ((1/16) x $win_sym) +
($cc_res **2) / ((1/16) x $win_sym))

push $chi_sq onto @chi_sq

105' shift (remove first element of an array and move all elements one step leftward) @trgt_fwd

push $basefeed onto array @trgt_fwd

106' assign $gg_count = 0, $ga_count = 0 ... one such variable for each of the 16 DPTs

assign index variable $i = 1

perform (win_sym) iterations:
- if the $i th elements of arrays @trgt_fwd and @trgt_revcomp are G and G, respectively, then increase $gg_count by 1. Repeat this conditional operation for each of the 16 possible DPTs
- increase $i by 1

calculate residuals $gg_res = $gg_count - ((1/16) x $win_sym) Repeat this calculation for each of the 16 possible DPTs.

push $gg_res onto @gg_res Repeat this step for each of the 16 possible DPTs

Calculate chi square. $chi_sq = (($gg_res **2) / ((1/16) x $win_sym) +
($ga_res **2) / ((1/16) x $win_sym) +
($gt_res **2) / ((1/16) x $win_sym) +
($gc_res **2) / ((1/16) x $win_sym) +
($ag_res **2) / ((1/16) x $win_sym) +
($aa_res **2) / ((1/16) x $win_sym) +
($at_res **2) / ((1/16) x $win_sym) +
($ac_res **2) / ((1/16) x $win_sym) +
($tg_res **2) / ((1/16) x $win_sym) +
($ta_res **2) / ((1/16) x $win_sym) +
($tt_res **2) / ((1/16) x $win_sym) +
($tc_res **2) / ((1/16) x $win_sym) +
($cg_res **2) / ((1/16) x $win_sym) +
($ca_res **2) / ((1/16) x $win_sym) +
($ct_res **2) / ((1/16) x $win_sym) +
($cc_res **2) / ((1/16) x $win_sym))

push $chi_sq onto @chi_sq

107 save @gg_res to file. Repeat this step for each of the 16 possible DPTs.

save @chi_sq to file

FIGURE 7

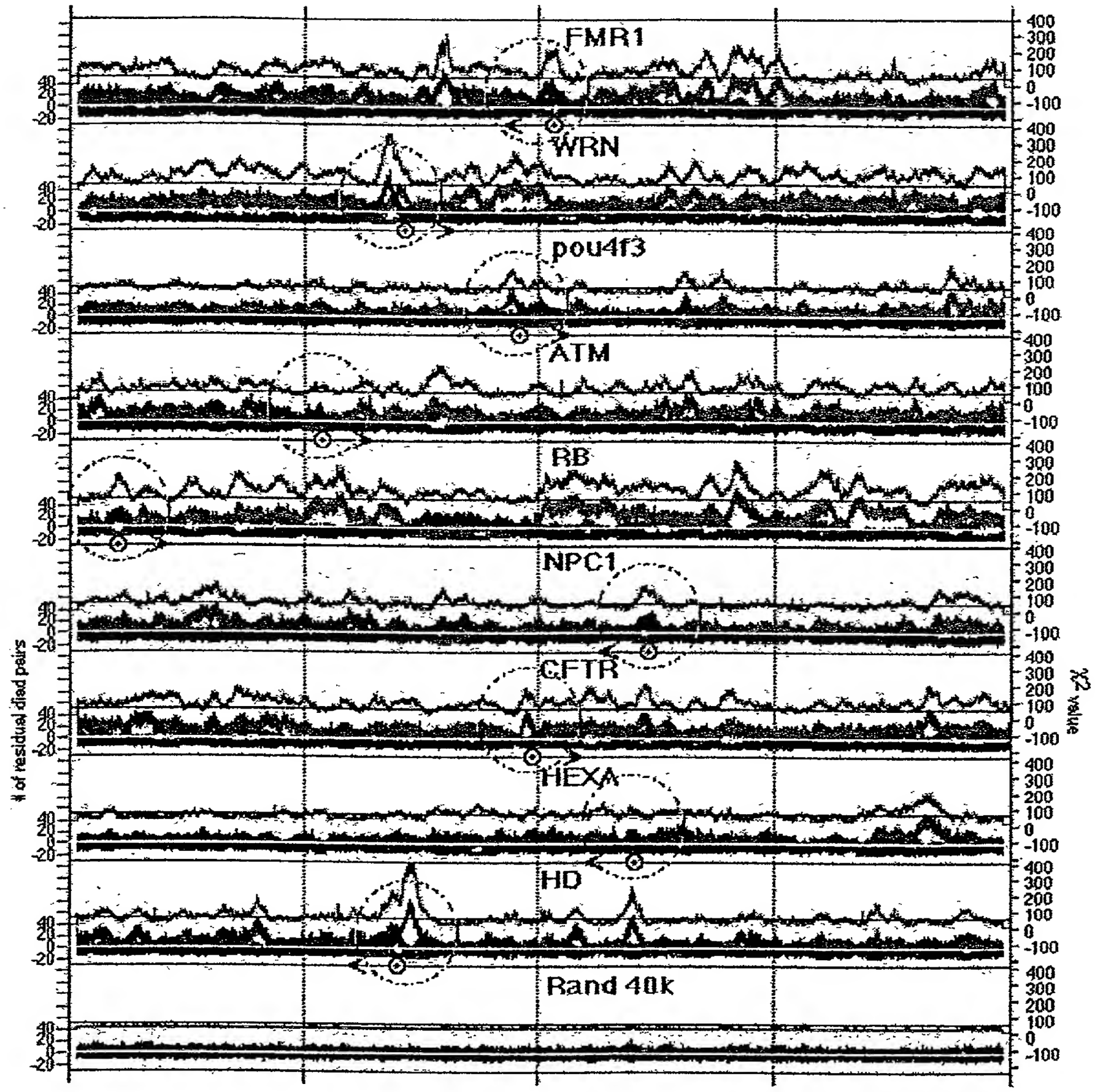
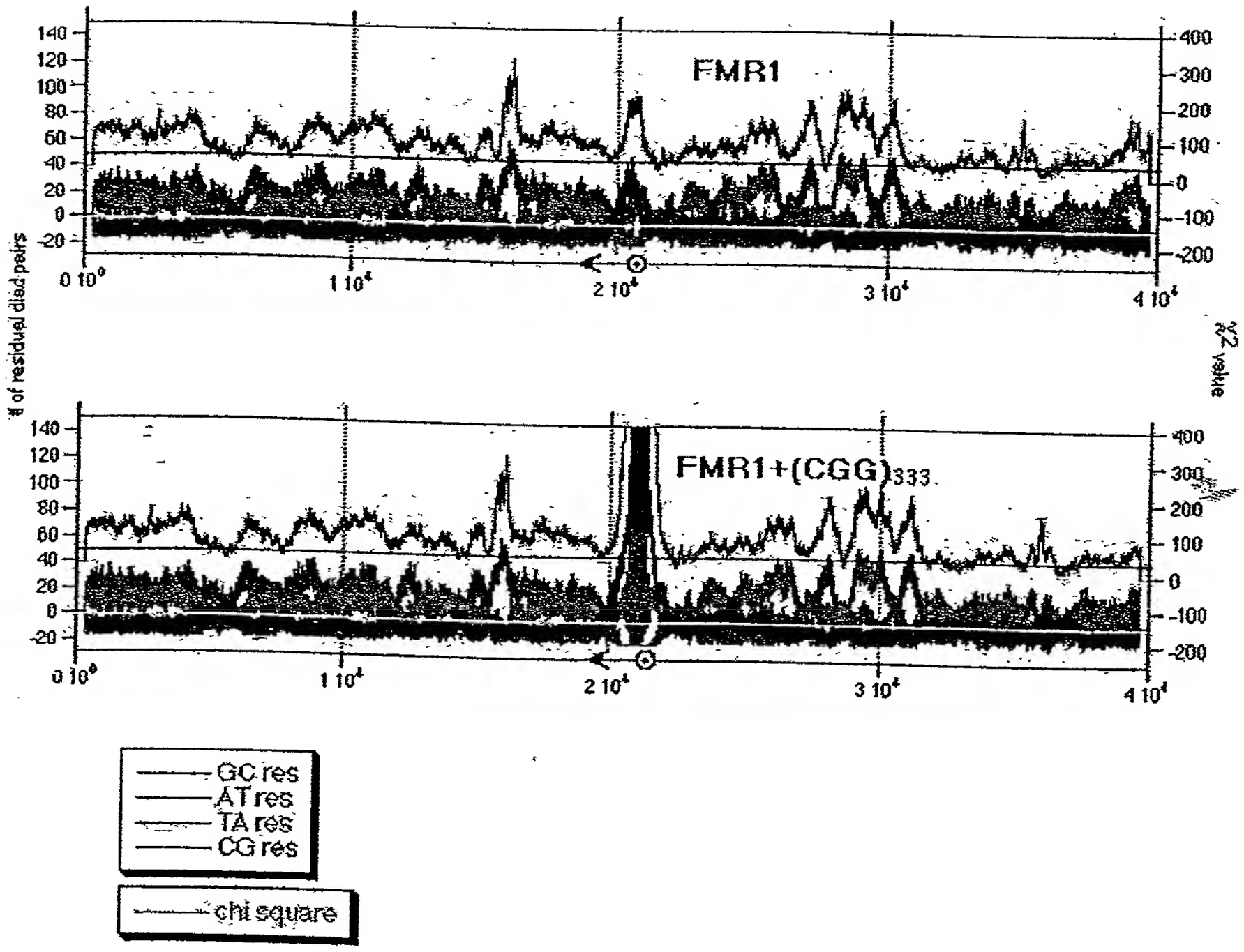

FMR1
WRN
pou4f3
ATM
BB
NPC1
CFTR
HEXA
HD
Rand 40k
# of residual diad pairs
χ2 value
GC res
AT res
TA res
CG res
chi square

FIGURE 8


FMR1
FMR1+(CGG)333
# of residual dsed pairs
value
0 10^0
1 10^4
2 10^4
3 10^4
4 10^4
GC res
AT res
TA res
CG res
chi square

A

B

FIGURE 9

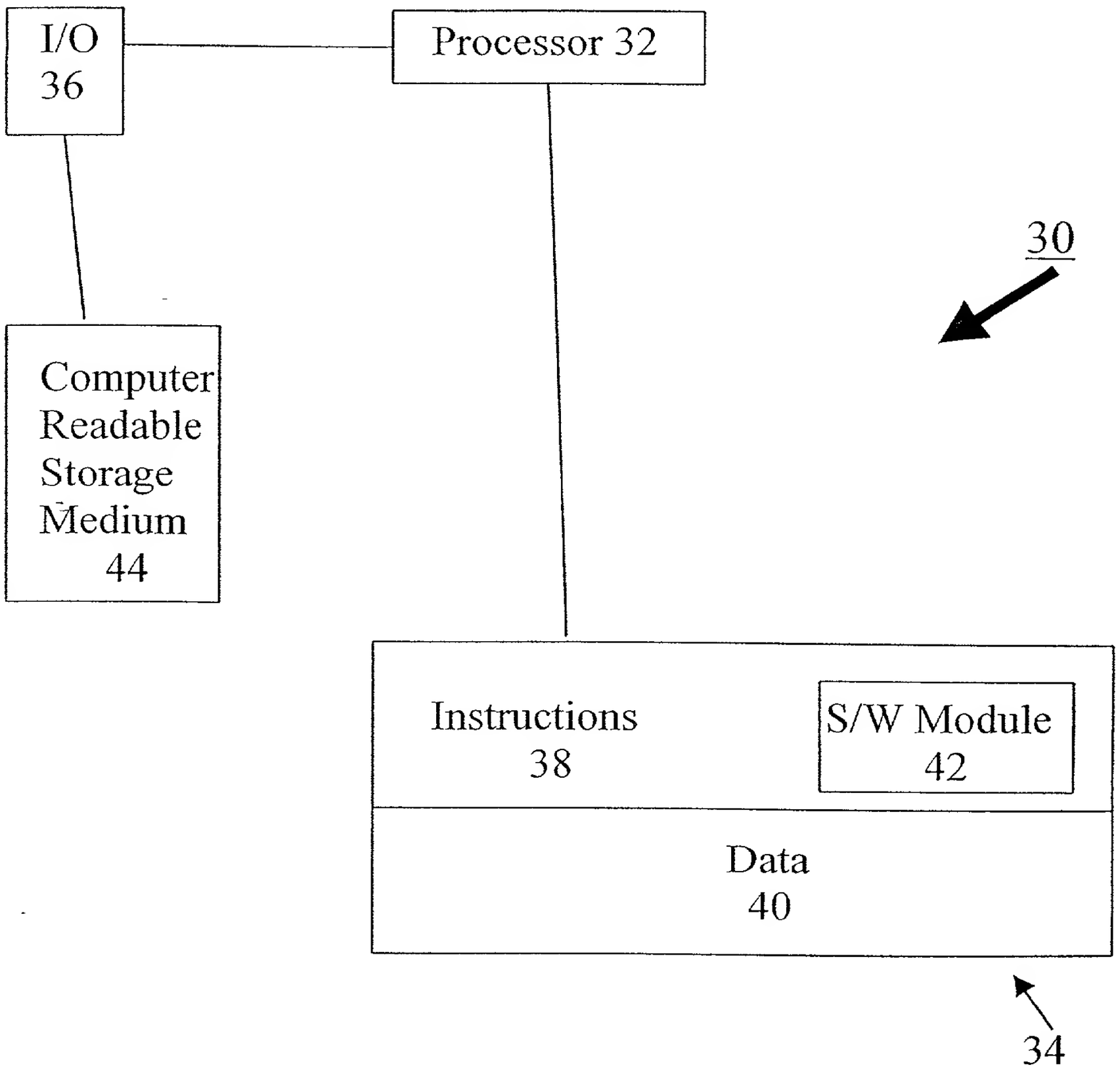

I/O
36
Processor 32
30
Computer
Readable
Storage
Medium
44
Instructions
38
S/W Module
42
Data
40
34

FIGURE 10